JTAG-based PLC Memory Acquisition Framework for Industrial Control Systems

By:
Muhammad Haris Rais (Virginia Commonwealth University), Rima Asmar Awad (Oak Ridge National Laboratory), Juan Lopez Jr (Oak Ridge National Laboratory), and Irfan Ahmed (Virginia Commonwealth University)

From the proceedings of

The Digital Forensic Research Conference

DFRWS USA 2021

July 12-15, 2021

JTAG-based PLC memory acquisition framework (Kyros) for industrial control systems

Muhammad Haris Rais, Rima Asmar Awad, Juan Lopez Jr. , Irfan Ahmed

Virginia Commonwealth University Oakridge National Laboratory

Industrial Control System

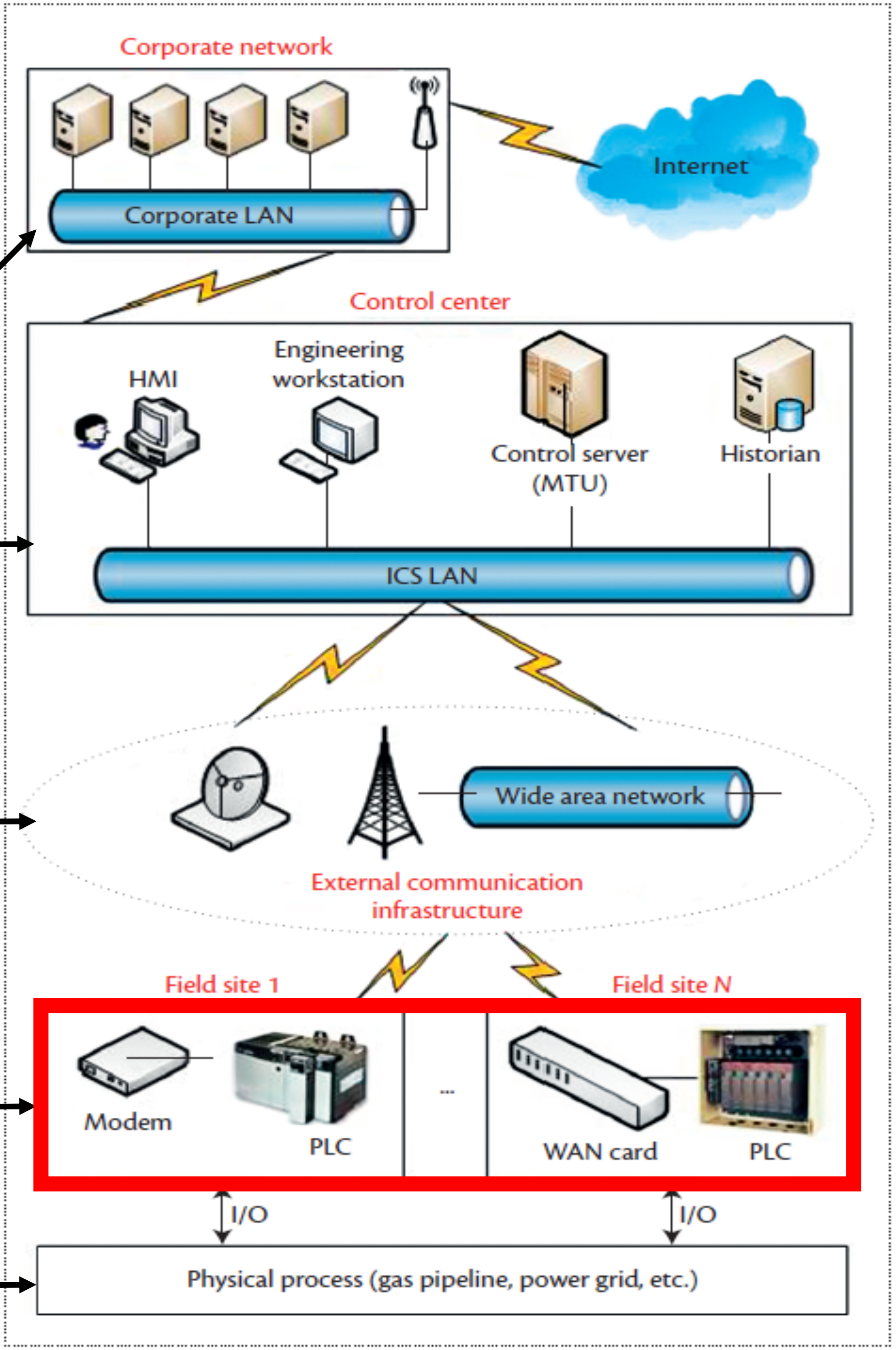
IT Network

Control Layer

WAN/ LAN / Other

Interfacing layer

Physical process layer

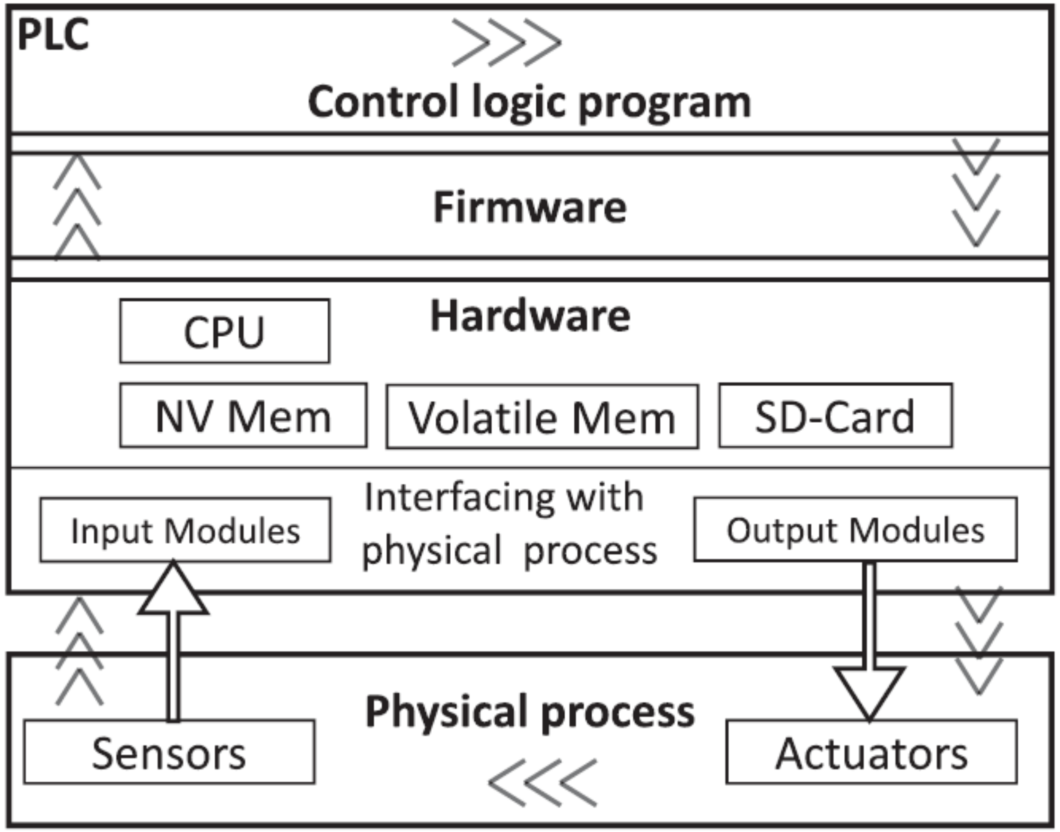


DFRWS

SAFE Lab

Programmable Logic Controller Architecture

- Get current state of system
- Apply the user-programmed logic
- Calculate the new output values
- Apply to actuators

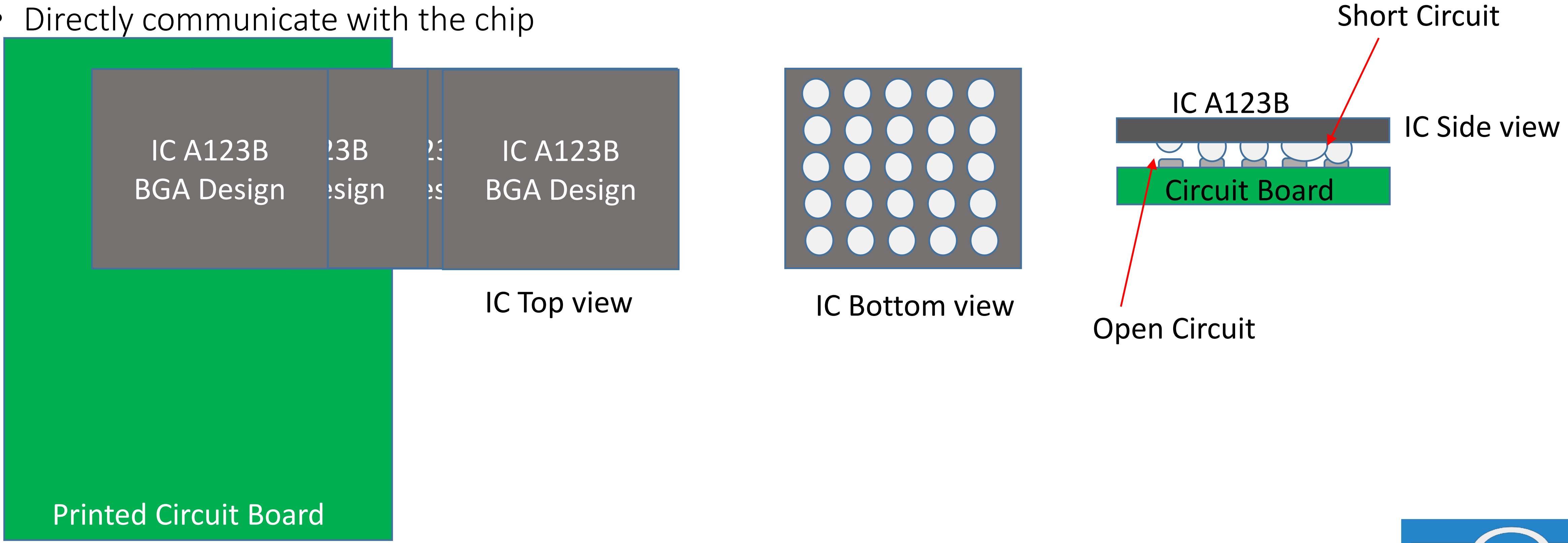


Memory Forensics of a PLC and its Challenges

- Two distinct research areas in memory forensics
  - Acquisition
  - Analysis
- PLC memory acquisition research is focused around
  - PLC debugging tools
  - Using ICS protocols (such as PCCC)
  - Network data
- If physical access is available, why not use a hardware based approach
- JTAG interface is explored in past for modifying PLC firmware
- No framework to guide about the memory extraction process

Joint Test Action Group (JTAG) IEEE 1149.1

- JTAG is a common name for IEEE standard for boundary scan architecture
- Initiated for validating the complex printed circuit board assembly
- Now used for testing, debugging, programming embedded systems
- Directly communicate with the chip

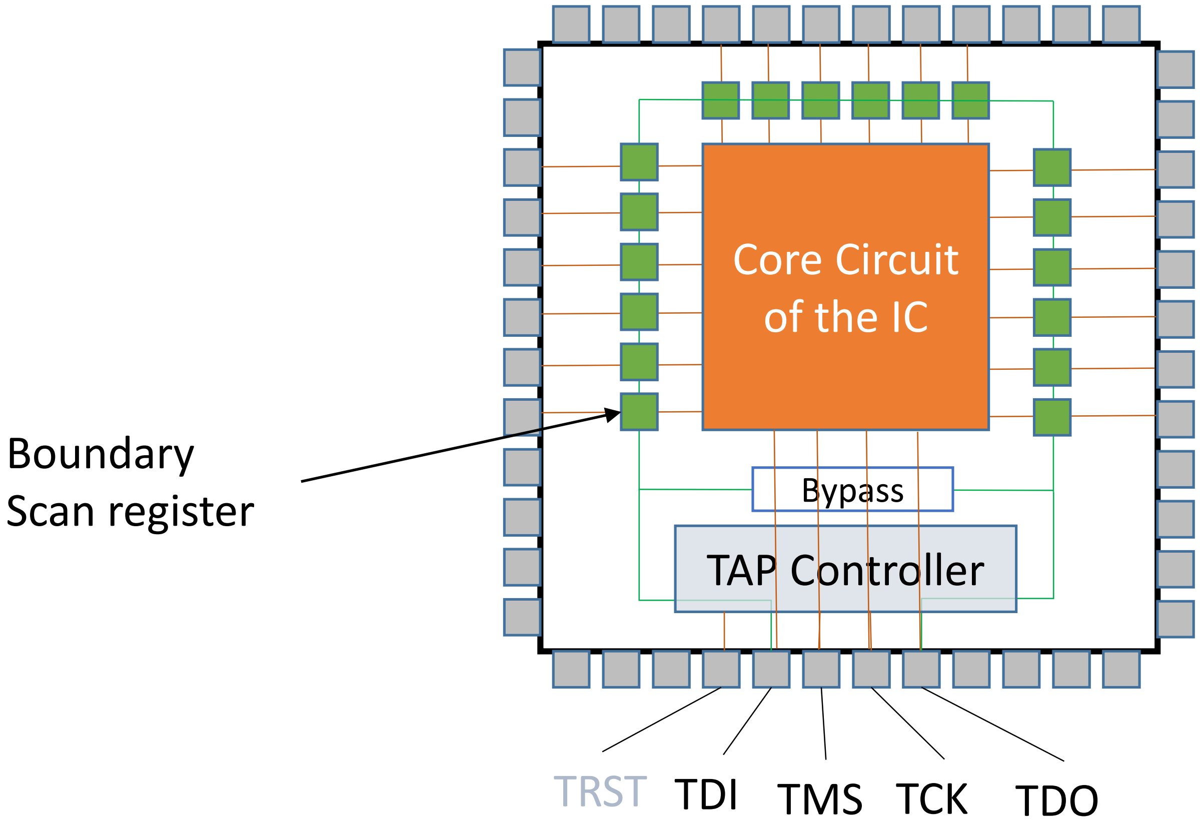


DFRWS

SAFE Lab

JTAG in the Integrated Circuit

- Test Access Point Controller – a simple state machine



JTAG PINS:

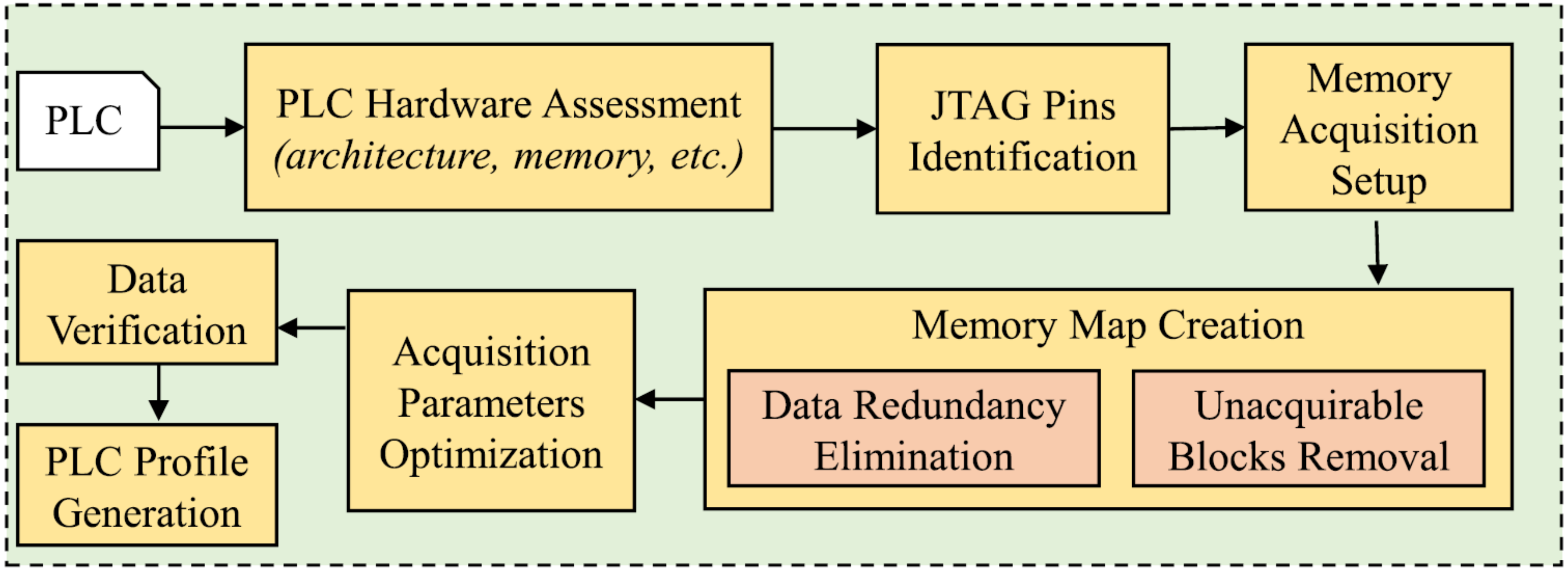
TDI: Test Data IN
TDO: Test Data OUT
TMS: Test Mode Selector
TCK: Test Clock
TRST: Test Reset

Kyros Framework for PLC Memory Acquisition through JTAG

Kyros Framework’s Phases

- Kyros comprises of 2 distinct phases
    1. PLC memory profile creation using a test PLC
    2. Memory acquisition of the suspect PLC

Memory Profile Creation Phase



Final Outcome of this phase

- JTAG setup details
- Memory address map of the PLC
- Memory acquisition parameters

Phase 2

DFRWS

SAFE Lab

Hardware Assessment

- Microcontroller
  - Architecture
  - Internal volatile and non-volatile memory elements
  - JTAG pins details
  - Memory address map
- Memory elements available
  - Volatile components
  - Non-volatile components
- Removable storage

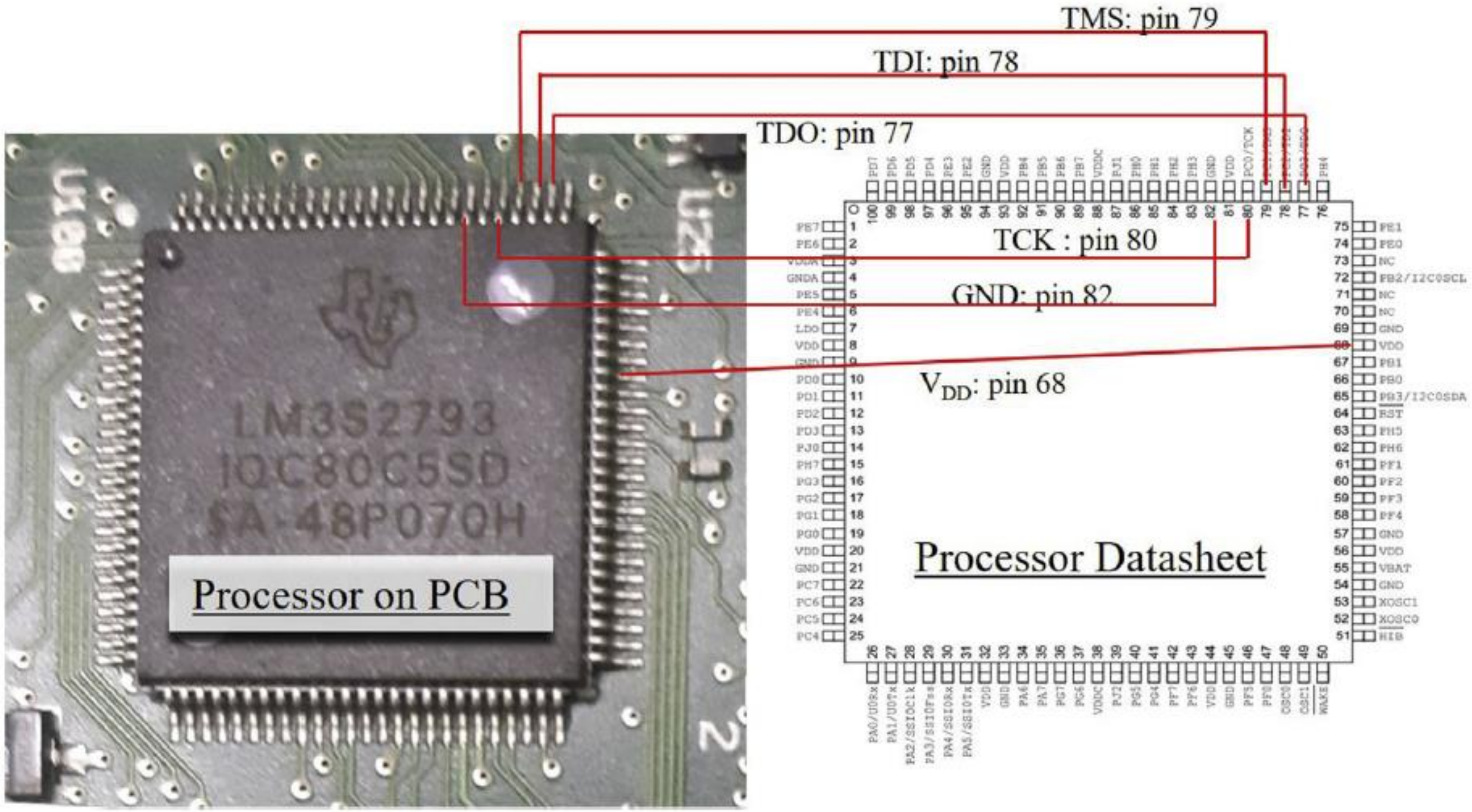
Start	End	Description
Memory		
0x0000.0000	0x0001.FFFF	On-chip Flash
0x0002.0000	0x00FF.FFFF	Reserved
0x0100.0000	0x1FFF.FFFF	Reserved for ROM
0x2000.0000	0x2000.FFFF	Bit-banded on-chip SRAM
0x2001.0000	0x21FF.FFFF	Reserved
0x2200.0000	0x221F.FFFF	Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000
0x2220.0000	0x3FFF.FFFF	Reserved
FiRM Peripherals		
0x4000.0000	0x4000.0FFF	Watchdog timer 0
0x4000.1000	0x4000.1FFF	Watchdog timer 1
0x4000.2000	0x4000.3FFF	Reserved
0x4000.4000	0x4000.4FFF	GPIO Port A
0x4000.5000	0x4000.5FFF	GPIO Port B

Hardware Assessment – Main Sources of Information

1. PLC hardware manuals
   - Code memory, IO memory, controller architecture and ID, backup options

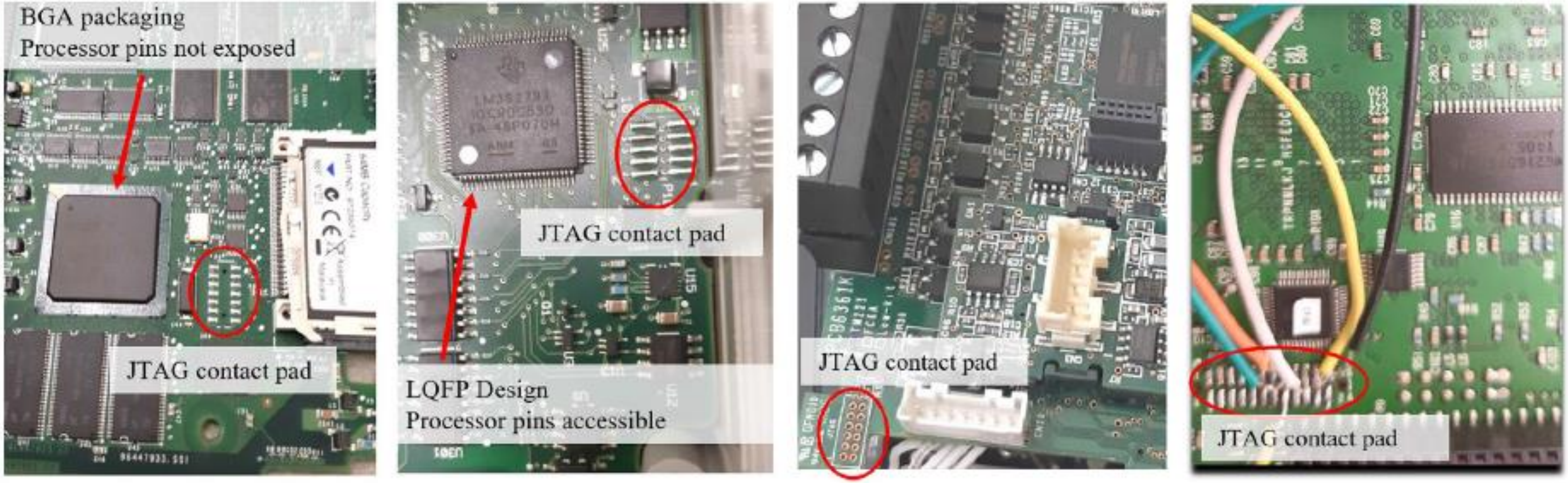
2. Visual inspection by disassembling
   - Main board, communication board, power board, IO boards
   - Examine the ICs on the circuit boards
     - Controller IC
     - Memory Elements
3. IC Datasheets
   - Processor IC
   - Memory IC type / capacity / parameters

Identifying JTAG pins from Processor datasheet



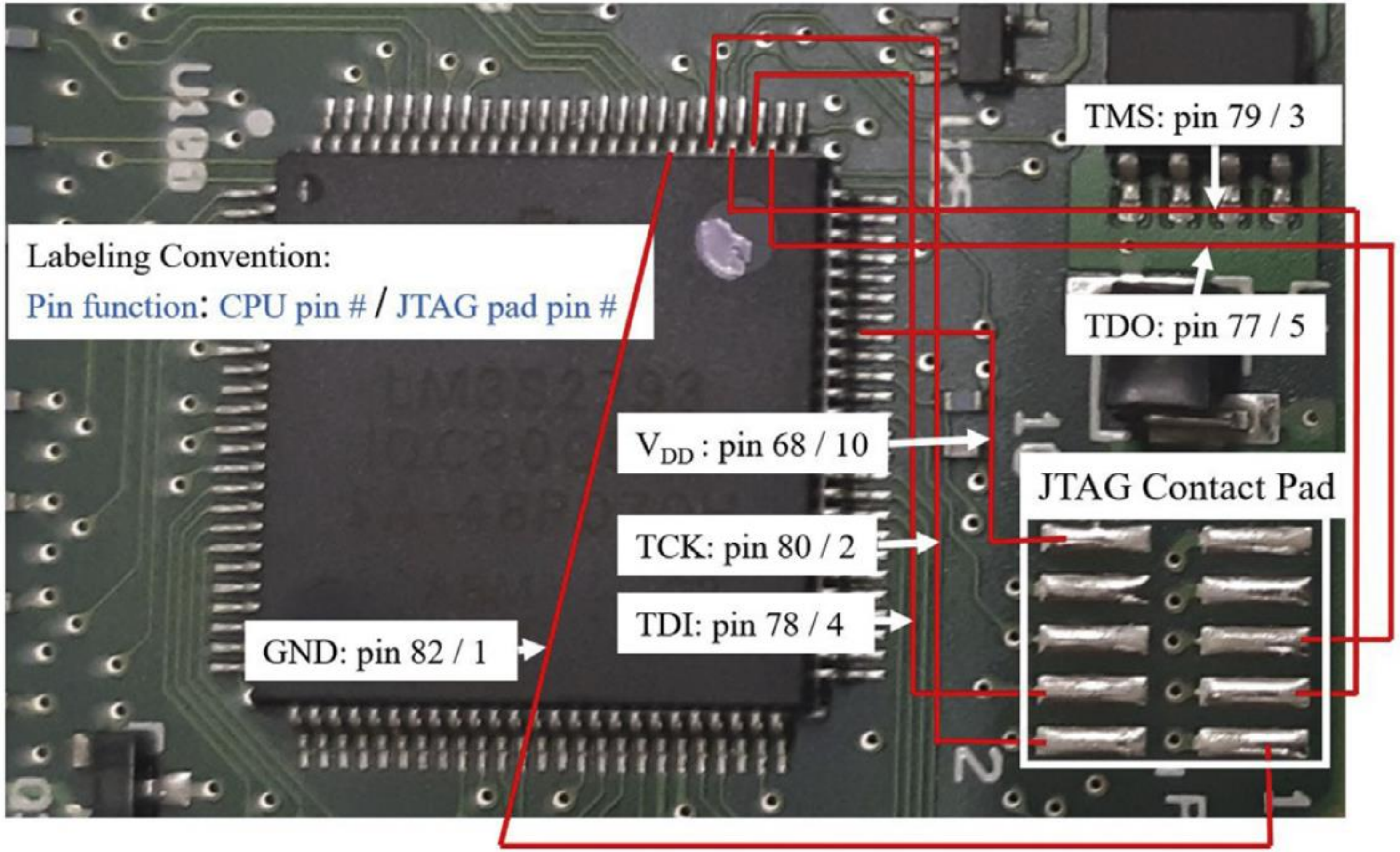
Identifying JTAG contact pad

- Direct access to the controller pins is difficult / risky or impossible in some cases
- Original JTAG contact pad is available on the circuit board
- Most likely, the connector will be removed
- Use heuristics, connectivity tests, trials to identify correct contact pad

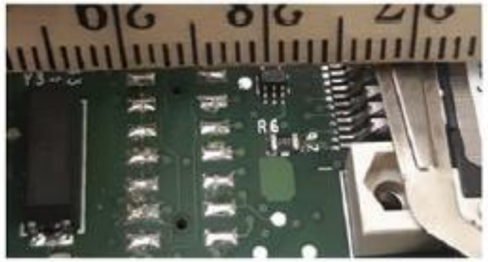


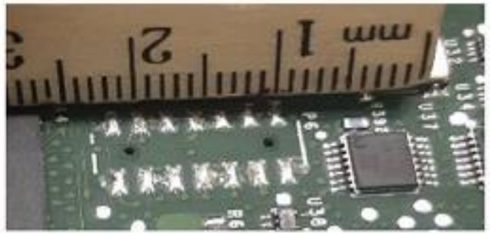
(a) ControlLogix 1756 (b) CompactLogix 1769 (c) Modicon M221 (d) MicoLogix 1100

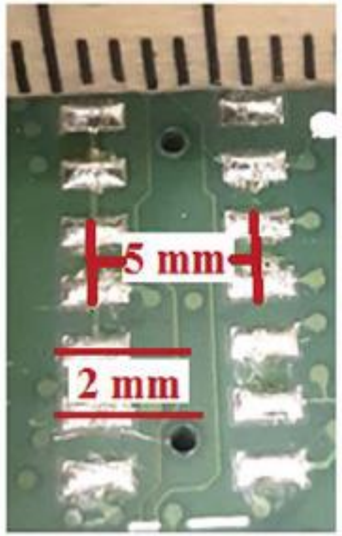
Finding JTAG contact pad through connectivity test

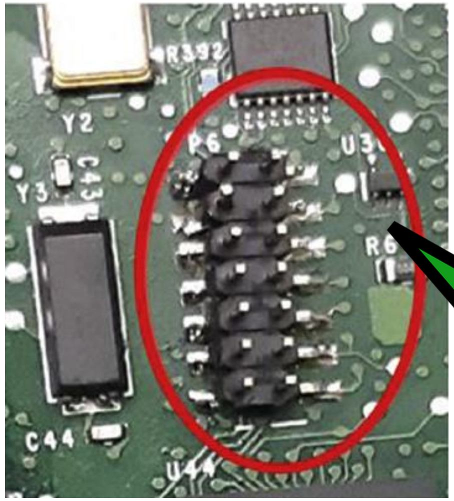


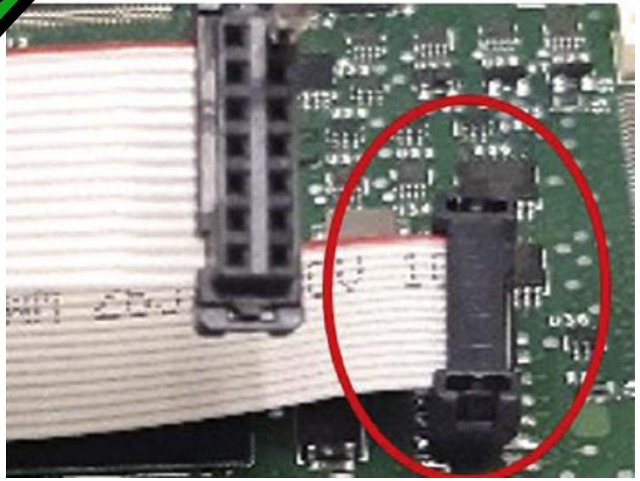
Header Installation on the JTAG pad

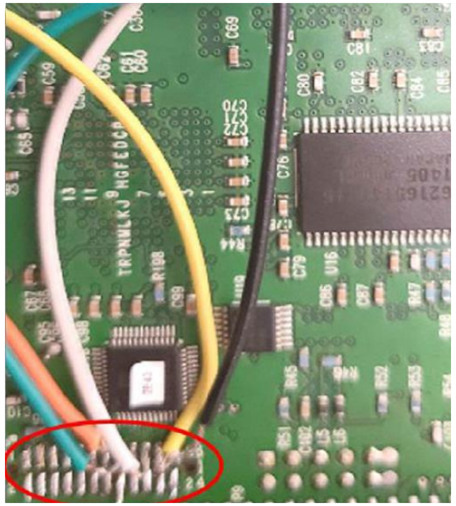








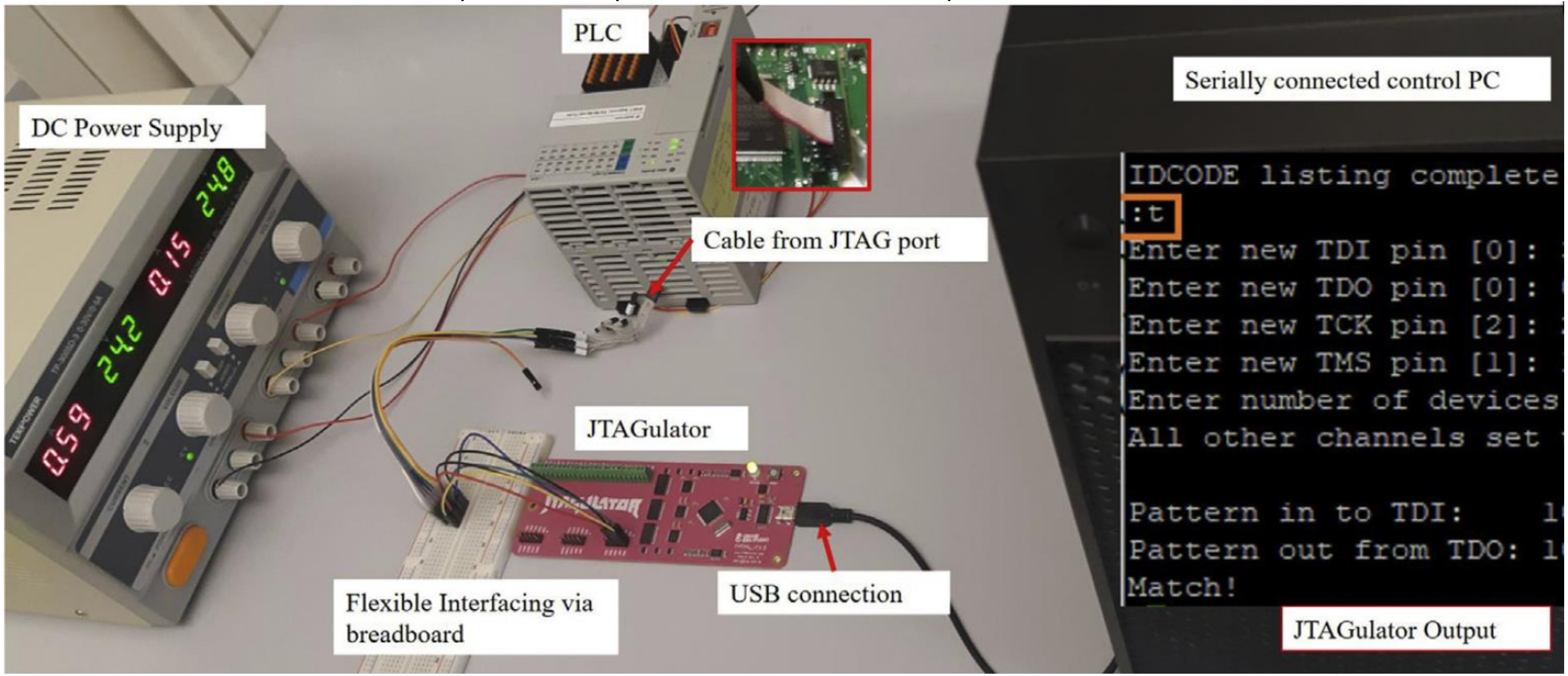




Soldering Wires on the JTAG contact pad

Ascertaining the JTAG status and confirming the pins

- Some vendors disable the JTAG port after testing the board
- Device like JTAGulator helps identify the status and the pins functional role



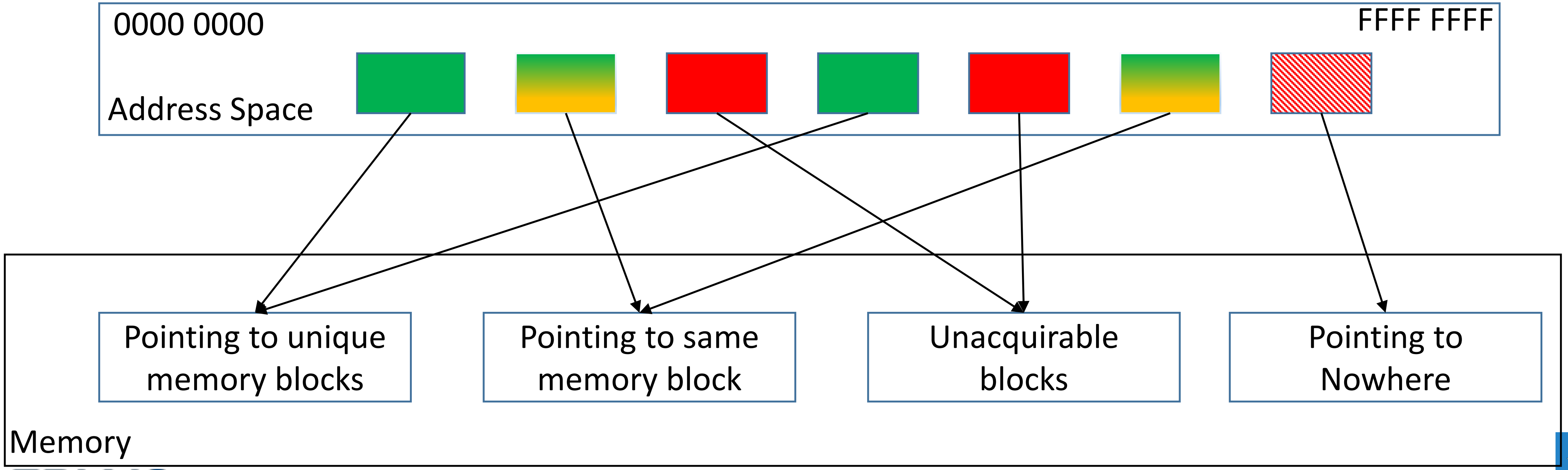
- Caution: Ground pin of Jtagulator and the circuit board should be correctly connected

Memory Acquisition Setup

- How to utilize JTAG for memory acquisition?
  - JTAG has TDI, TDO, TMS, TCK pins working at the controller’s operating voltage
  - We need voltage converter to adapt to the target controller
  - Need processor’s BSDL file and create program to exploit JTAG for reading memory
- Simpler option is to use an off-the-shelf JTAG debugger supporting the processor’s architecture
- Some famous debuggers include
  - Segger Jlink
  - Atmel-ICE
  - Renesas E2

Memory Map Creation

- Memory acquisition through JTAG is slow and risky process
- Typically, PLC uses a small fraction of complete address space (ie 4GB for 32 bit processors)
- Eliminate
  - the unused address ranges
  - the redundant address blocks
  - the unacquirable address space



Caution while eliminating data redundancy

- Not all duplicate data blocks are redundant
- PLC may keep multiple copies of data
- For example, firmware is loaded from flash to RAM
- Both are independent artifacts
- Confirm before eliminating
- For instance,
  - Consider the address boundaries
    - If no unused pins identified, keep the copies
  - Write to an address to see effect on other copies

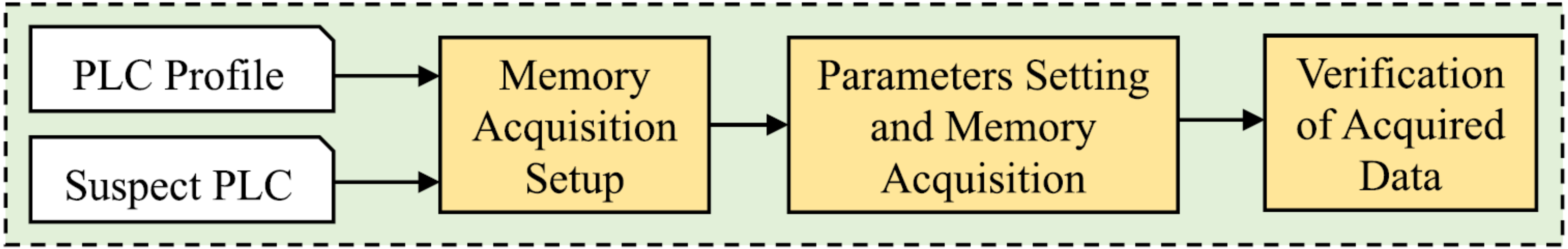
Eliminating the Unacquirable Address Blocks

- Address blocks allocated to hardware devices / peripherals
- Attempt to read may hang or crash the PLC
- Check processor datasheets
- Use “Crash and Learn” exercise during initial acquisition on test PLC
- Employ software based device recovery technique to avoid manual intervention
    - Using PLC API
    - Resetting power supply through software
- Debugger’s crash events can be handled conveniently

Acquisition Parameters Optimization

- JTAG based acquisition is below firmware level
- May expire some watchdog timers resulting in PLC crash
- Optimization also helps in speeding up the process
- On a per address block basis
- Important parameters to optimize
  - Acquisition speed– the clock rate of debugger connected to PLC via JTAG
  - Block size - Memory block to read under single command
  - Debugger’s buffer size
  - Wait time between consecutive read operations
- Parameters can be found
  - Theoretically derive through the information of processor, IC chips, and the debugger
  - Practically discover through a “ramp up till crash” approach
- With limited memory regions, 2nd approach is faster

Memory Acquisition of the Suspect PLC



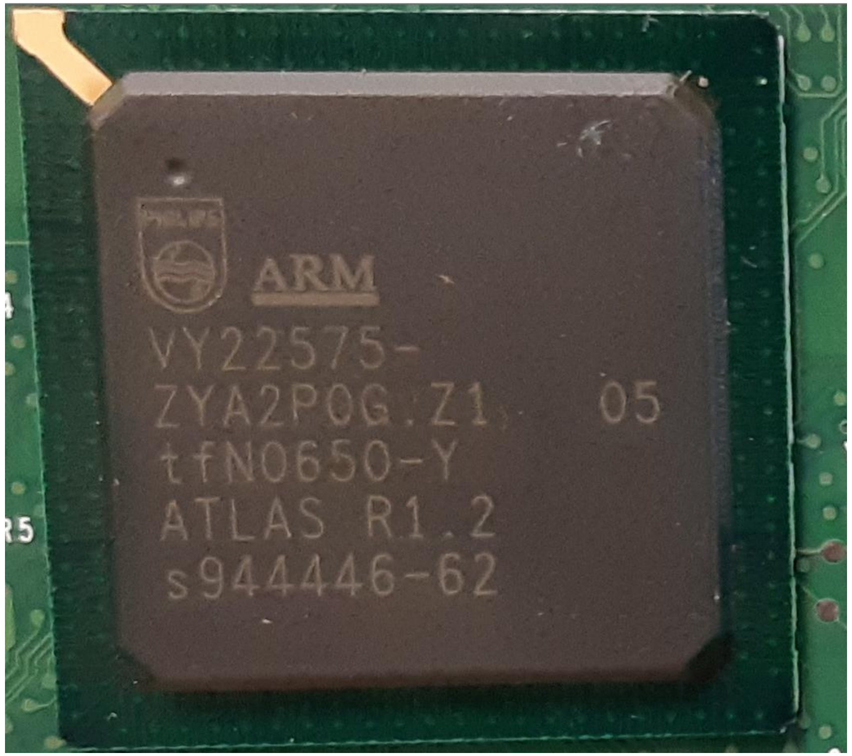
<u>Data Verification</u>

- Verifying firmware from vendor website / SD card
- Known configured data
- Writing and reading back

CASE STUDY ON ALLEN BRADLEY CONTROL LOGIX 1756 A/10 WITH 1756-L61 CONTROLLER

Hardware Assessment - Micorcontroller

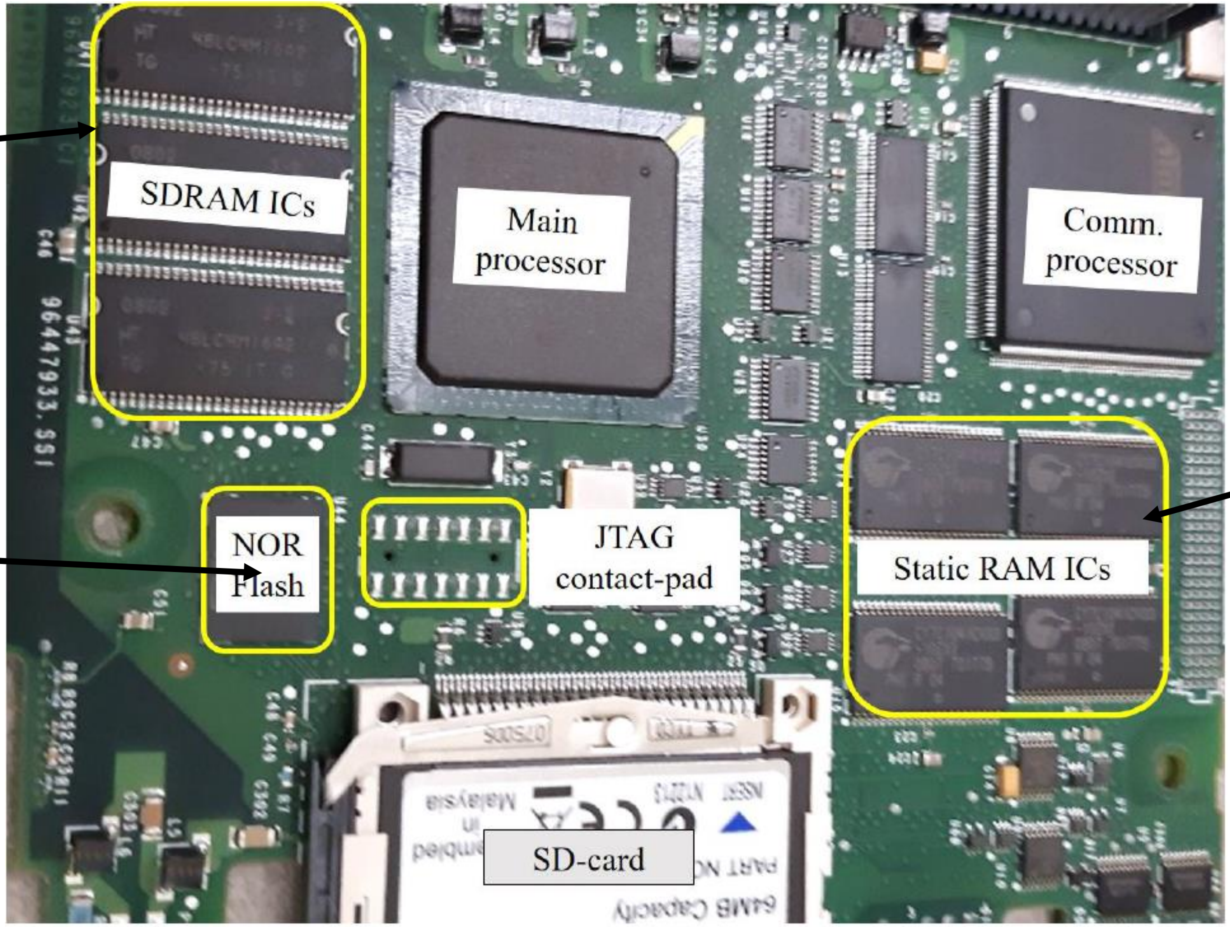
- Chassis populated with Controller, Input, Output, Communications, Counter modules
- No details about processor found in datasheets
- Disassembly of controller module 1756-L61
  - Main processor Philips VY22575 (centrally located)
  - No documentation found
  - NXP regretted to provide information (being customer specific IC)
  - ARM processor



Hardware Assessment – Memory Elements

8 MB each

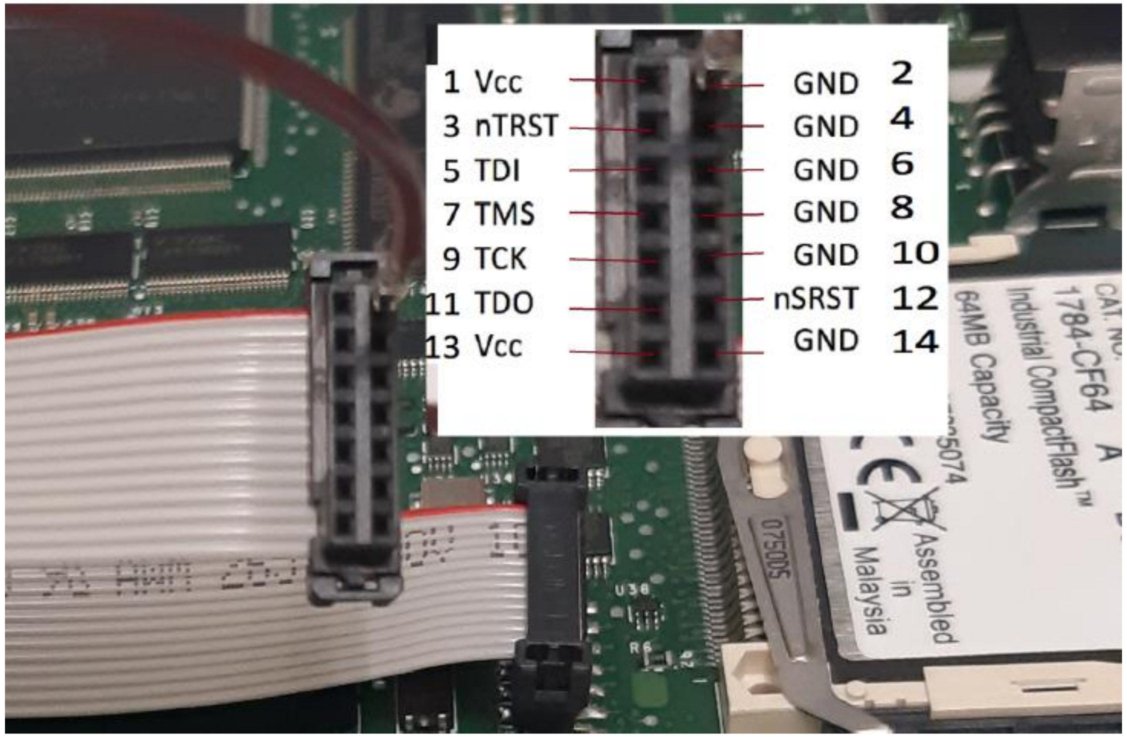
8 MB



512 KB each

JTAG pins identification

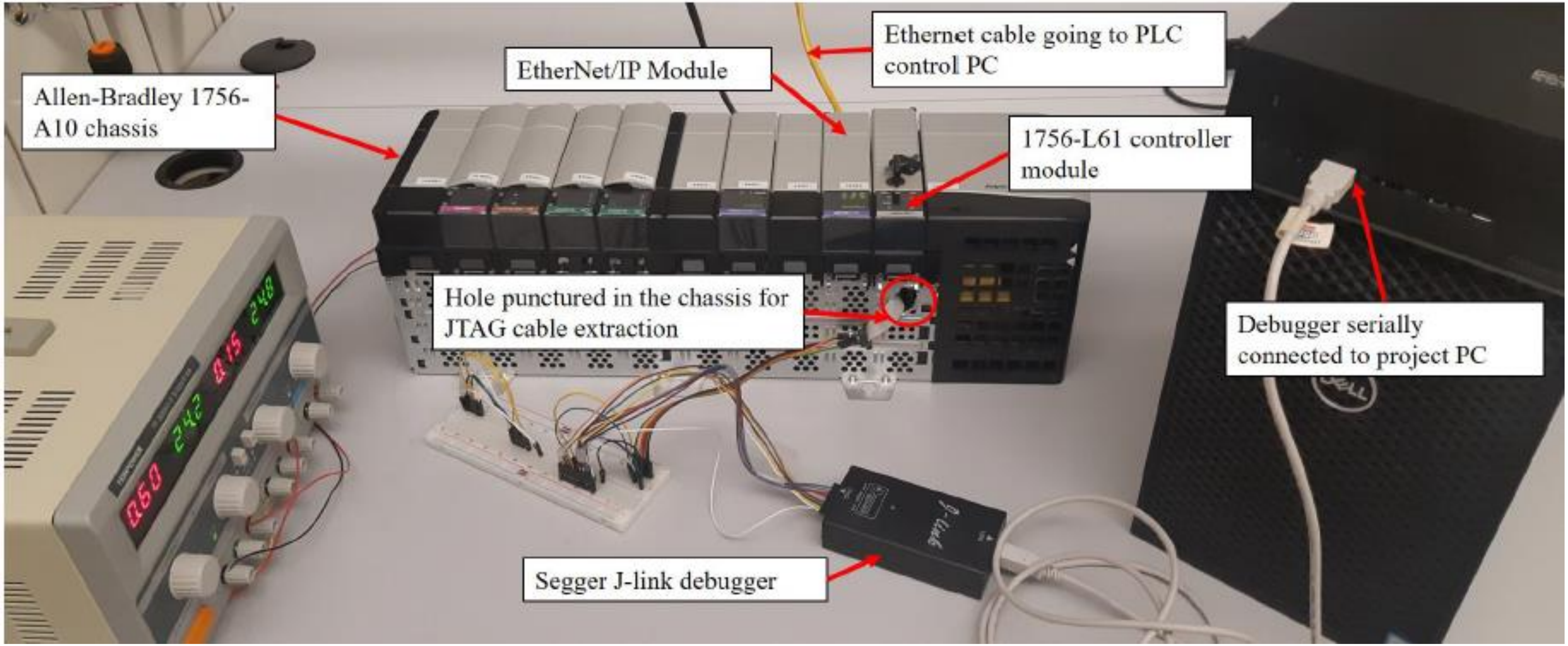
- A 2x7 pins contact pad in vicinity of processor
- Connectivity test not possible
  - BGA Design
  - No datasheets available
- Header installed on the contact pad
- JTAGulator confirmed the JTAG pins and JTAG status as working



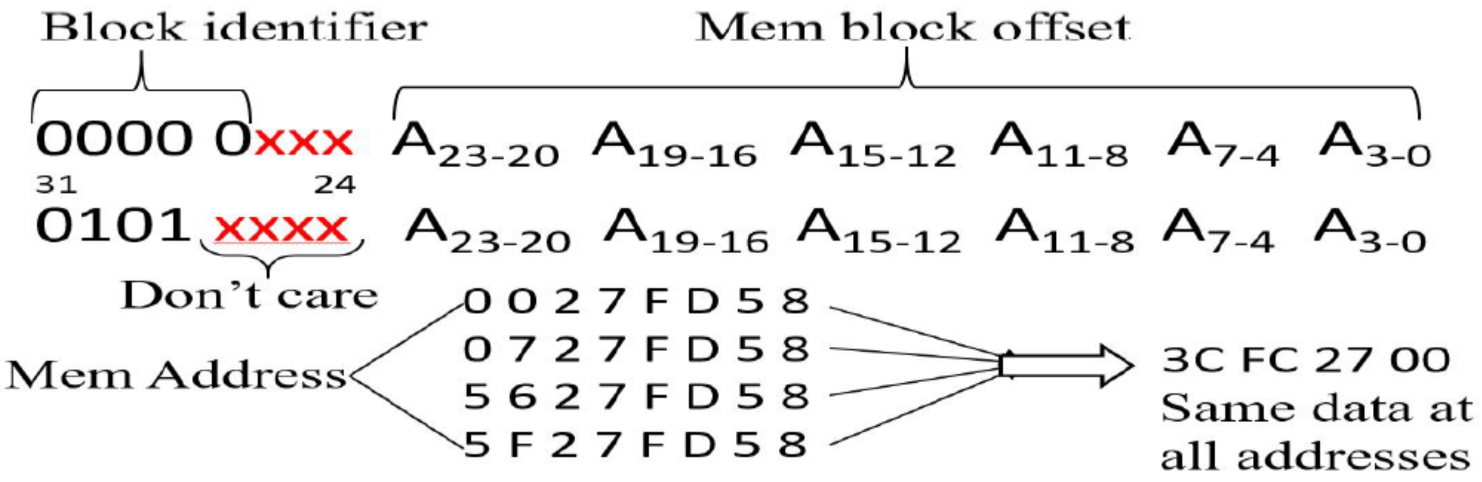
DFRWS

SAFE Lab

Memory Acquisition Setup



Redundant Data Case: Unused Address Pins Example



Parameters Optimization

- 28 distinct address ranges discovered
- Applied ramping up technique for parameter optimization
- Different regions offer different values of parameters
- Run vs Program mode
- Marker based acquisition

Starting Address	Ending Address	Block Size	Speed	Wait Time	Marker Based
'00000000',	'0007FFFF' ,	1024 ,	0 ,	4 ,	0
'00080000',	'0027FFFF' ,	2048 ,	0 ,	4 ,	0
'08010E00',	'08010EFF' ,	64 ,	0 ,	0 ,	0
'0A000000',	'0A7FFFFF' ,	8192 ,	0 ,	4 ,	0
'0C000000',	'0C003FFF' ,	64 ,	0 ,	4 ,	0
'0C004000',	'0C07F7FF' ,	128 ,	0 ,	5 ,	0
'0C07F800',	'0C07FFFF' ,	1024 ,	0 ,	1 ,	0
'0C080000 ,	'0C081F0F'	immediately halts			
'0C081F10',	'0C08283F' ,	64 ,	0 ,	0 ,	0

Finalized Address Ranges and Optimized Parameters

```
= [ [ '00000000', '0007FFFF' , 1024 , 0 , 4 , 0 ] ,   # Part 1 of 16MB repeated  8 times, and covers till 7FFF FFFF #  Split into 3 parts
    [ '00080000', '0027FFFF' , 2048 , 0 , 4 , 0 ] ,   # Part 2 of 16MB: Data and logic zone : 2MB in total but due to markers, we extract
    [ '00280000', '00FFFFFF' , 8192 , 0 , 5 , 0 ] ,   # Part 3 of 16MB
    # 01000000 to 07FFFFFF  Repeatition of the above 16MB (total 8 copies of above data)
    [ '08000000', '08000FFF' , 64 , 0 , 0 , 0 ] ,  # 4KB - 0 [8 or 9] X  x x x 0   X A B C
    [ '08010080', '0801037F' , 64 , 0 , 0 , 0 ] ,  # 2nd 4KB first piece ; gaps not readable - 0 [8 or 9] X  x x x 1   X D E F
    [ '08010400', '0801047F' , 64 , 0 , 0 , 0 ] ,  # 4KB in pieces ; gaps in these ranges not readable - 0 [8 or 9] X  x x x 1   X D E F
    [ '08010580', '080108FF' , 64 , 0 , 0 , 0 ] ,  # 4KB in pieces ; gaps in these ranges not readable - 0 [8 or 9] X  x x x 1   X D E F
    [ '08010980', '08010AFF' , 64 , 0 , 0 , 0 ] ,  # 4KB in pieces ; gaps in these ranges not readable - 0 [8 or 9] X  x x x 1   X D E F
    [ '08010C80', '08010D7F' , 64 , 0 , 0 , 0 ] ,  # 4KB in pieces ; gaps in these ranges not readable - 0 [8 or 9] X  x x x 1   X D E F
    [ '08010E00', '08010EFF' , 64 , 0 , 0 , 0 ] ,  # 4KB in pieces ; gaps in these ranges not readable - 0 [8 or 9] X  x x x 1   X D E F
    [ '0A000000', '0A7FFFFF' , 8192 , 0 , 4 , 0 ] ,   # 8MB repeated 4 times; covers till 0BFF FFFF
    [ '0C000000', '0C003FFF' , 64 , 0 , 1 , 0 ] ,
   [ '0C004000', '0C07F7FF' , 4 , 0 , 4 , 0 ] ,  # IO DATA; to be fetched really slow; markers 80000001; from start and end
    [ '0C07F800', '0C07FFFF' , 64 , 0 , 1 , 0 ] ,
    # 0c0800000 , 0c081f0f immediately hangs the processor [0C and 0D has same data
    [ '0C081F10', '0C08283F' , 64 , 0 , 0 , 0 ] ,  #
    [ '0E000000', '0E000FFF' , 256 , 0 , 0 , 0 ],    # Fast counters with dominently zeros; not sure of the boundary (seems few KB)- same
    [ '10000000', '10000FFF' , 256 , 0 , 0 , 0 ],    # Fast changing data packed with FFFFs. Size not sure - repeated till 13FFFFFF
     # '14000000', '17FFFFFF' no data - debugger hangs - seems that region is not assigned


    [ '60002000', '600027FF' , 64 , 0 , 0 , 0],
    # 2KB data (changeable; 2 bits dont care matching the 8KB above and making a block of 16KB from 60000000 to 60003FFF - this covers till
    # 68000000 onwards not readable and/or not configured ; debugger returns Nil data
```

DFRWS

SAFE Lab

Phase2: Memory Acquisition of suspect PLC

- Due to non-availability of another PLC of same model, we used test PLC for the 2nd phase
- Python code utilizing Pylink library to interface with Segger Jlink debugger
- Program allow for 3 modes of acquisition
    - Complete acquirable memory
    - Acquisition of customized range
    - Signature-based acquisition

Data Verification

- Reasonable confidence already attained during profile creation
  - For instance, redundant data block checking process confirms the acquisition correctness
- Downloaded the firmware (2.7MB file)
  - Perfectly matched to the acquired multiple instances
- Acquisition of known data across the memory over multiple restarts

Limitations and Future Work

- Requirement of a separate PLC for memory profile creation
- Hardware interference – disassembly, soldering, header installation and cable routing
- Does not work if JTAG port is disabled by the vendor

- Employ and evaluate *kyros* framework on other PLCs

Acknowledgement

This work was supported, in part, by the Virginia Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation, and workforce development.

Conclusion

- We presented *Kyros* framework for PLC memory acquisition through JTAG interface
- *Kyros* guides a forensic researcher on
  - How to tackle the hardware challenges related to JTAG
  - How to deal with proprietary hardware and custom-made IC with no access to datasheets
  - How to generate a memory map and optimize the acquisition parameters
- We presented a case study of the framework on Allen-Bradley ControlLogix 1756

THANKS !

Authors' Contact Info

Muhammad Haris Rais	raismh@vcu.edu
Rima Asmar Awad	awardrl@ornl.gov
Juan Lopez Jr.	lopezj@ornl.gov
Irfan Ahmed	iahmed3@vcu.edu